VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Matěj Korytář

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH A KONSTRUKCE MIDI PŘEVODNÍKU PRO SNÍMAČE STRUN
DESIGN AND CONSTRUCTION OF MIDI INTERFACE FOR SENSOR STRINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matěj Korytář**
AUTHOR

VEDOUCÍ PRÁCE — **MgA. Mgr. Ondřej Jirásek, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Matěj Korytář ***ID:*** 174456
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Návrh a konstrukce MIDI převodníku pro snímače strun

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a zkonstruujte MIDI převodník odečítající buď napětí ze snímačů, tlak z padů anebo přerušení laseru ze snímací části elektronického hudebního nástroje, tzv. cimbalonu, který substituuje struny a korpus cimbálu. Signál ze snímačů bude převodník převádět na zprávy Nota zapnuta a Nota vypnuta s informací o dynamice hraní, pokud to bude možné.

DOPORUČENÁ LITERATURA:

[1] GEIST, B. Akustika - jevy a souvislosti v hudební teorii a praxi. Praha: MUZIKUS s.r.o., 2005. ISBN 978-8086253312.

[2] SYROVÝ, V. Hudební akustika. Praha: AMU, 2003. ISBN 978-80-7331-127-8.

[3] FURIA, S., SCACCIAFERRO, J. The MIDI Implementation Book. Third Earth Production, Rutheford, 1988.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: MgA. Mgr. Ondřej Jirásek, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

Abstrakt

Tato práce se zaměřuje na problematiku návrhu a konstrukce MIDI převodníku pro hudební nástroj cimbál. Je zde stručně popsán akustický hudební nástroj z hlediska vzniku tónu a konstrukce. Dále jsou rozvedeny možnosti snímání strun, popřípadě jiné způsoby snímání hry. V následující kapitole je popsán MIDI protokol z hlediska historie, rozhraní a typů vysílaných zpráv. Jsou zde také uvedeny požadavky na čip, použitý pro převodník a způsob jeho připojení. Dále je popsán princip převodu signálu ze snímačů a následné zakódování do protokolu MIDI včetně použitého programu s detailním popisem. Na závěr je prodiskutováno několik budoucích rozšíření.

Klíčová slova

cimbál, MIDI protokol, MIDI převodník, mikročip, snímač

Abstract

This work focuses on problematics of MIDI converter design and construction for cimbalom. There is briefly described acoustic musical instrument in terms of tone creation and construction. Further is discussed possibilities of string sensing or other ways to capture performance. In the following chapter is described MIDI protocol in terms of history, interface and types of transmitted messages. There is also said requirements for converter's chip and way to connect it. Further is described a method of converting signal from sensors and following coding to MIDI protocol including used program with detailed specifications. In conclusion is discussed several future extension.

Keywords

cimbalom, MIDI protocol, MIDI converter, microchip, sensor

Bibliografická citace:

KORYTÁŘ, M. Návrh a konstrukce MIDI převodníku pro snímače strun. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 47 s. Vedoucí práce: MgA. Mgr. Ondřej Jirásek, Ph.D.

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma Návrh a konstrukce MIDI převodníku pro snímače strun jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne **5. června 2017** ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce MgA. Mgr. Ondřeji Jiráskovi, PH.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne **5. června 2017** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Cílem této práce je navrhnout a zkonstruovat MIDI převodník, který odečítá a zpracovává hudební informace při hře na elektronický hudební nástroj zvaný cimbalom neboli cimbál. Tyto informace budou dále vysílány do virtuálního programu simulujícího zvuk živého cimbálu. V práci bude probráno několik možností snímání strun, nebo jiného zdroje impulzu či vibrací, princip jejich převodu na elektrický signál a následný převod do digitální podoby. Bude zde také rozebrána problematika rozhraní MIDI, její fyzická vrstva, způsob komunikace a možnosti posílání dalších údajů mimo informací o aktivovaných tónech. Další část bude obsahovat výběr vhodného jednočipového procesoru, s ohledem na požadovaný počet vstupů a výstupů a rychlosti zpracování informací.
Na základě znalostí problematiky snímání úderů a komunikačního rozhraní MIDI bude rozebrán způsob řešení převodníku a následný popis programu pro jednočipový procesor.
Závěrem této práce bude funkční MIDI převodník s návrhy pro budoucí rozšíření, jako je například možnost fyzického ovládání některých parametrů virtuálního hudebního nástroje přes MIDI protokol.

2 HUDEBNÍ NÁSTROJ CIMBÁL

Dříve než začneme navrhovat způsob snímání, musíme se nejprve seznámit s konstrukcí samého hudebního nástroje, který vydává zvuk na základě mechanického principu. Jeho konstrukce je velmi podobná konstrukci klavíru, jen místo klaviatury s kladívkovou mechanikou používá hráč k vybuzení struny paličky (excitátor), jejichž hrací část bývá nejčastěji obalena vatou nebo plstí. Struny (oscilátor) nejsou rozmístěny v rámci jednoho rámu od nejdelších s nejnižším tónem po ty nejkratší s nejvyšším tónem jako u klavíru, ale z prostorových důvodů tak, aby se všechny vešly do lichoběžníkovitého korpusu. Jako zesilovač chvění strun (rezonátor) a zároveň část která zvuk vyzařuje (radiátor) slouží rezonanční deska. Celkový tónový rozsah je od velkého C po a3, tedy 58 tónů což dává dohromady téměř 5 oktáv. [1, 2]

Obr. 2.1 Hudební nástroj cimbál [2]

V případě konstrukce elektronického cimbálu by se mělo zamezit vlastnímu akustickému vzniku tónu tak, aby nerušil generovaný zvuk z reproduktorů. Toho dosáhneme buď zamezením přenosu vibrací na rezonanční desku, popřípadě současným nahrazením rezonanční desky materiálem, který rezonance utlumí. Dalším způsobem řešení je nahradit kmitací struny jiným prvkem, který zvuk nevydává.

3 ZPŮSOBY SNÍMÁNÍ A NÁVRH PŘEVODU DO DIGITÁLNÍ PODOBY

Existuje několik druhů, jak získat informace o hře na hudební nástroj. Zde jsou pro názornost uvedeny čtyři varianty. Pro každou je vysvětlen princip a možný způsob zpracování signálu.

3.1 Odečítání napětí ze snímačů

Tento způsob snímání se podobá principu elektrické kytary. Struna sama o sobě vibruje ale vibrace se nepřenáší dál a nezesilují. Pro každý tón by se musel použít samostatný elektromagnetický snímač, což dává v součtu 58 snímačů. Výhodou je, že na jedné struně vzniká pouze jeden tón určité výšky. Stačilo by tedy jen odečítat napětí ze snímačů a toto napětí měřit. Čím větší úder, tím větší by byl impulz a tím větší napětí. K tomuto způsobu bychom však potřebovali ideálně 58 A/D převodníků, abychom mohli přenášet dynamiku hry. Nevýhodou by byly možné přeslechy mezi těsně umístěnými strunami a nutnost přizpůsobení úrovně ze snímačů do převodníků. Šlo by tedy o náročný a finančně nákladný způsob snímání, na druhou stranu hráčsky nejpřívětivější, protože by se nelišil od klasického cimbálu. Popis magnetického snímače viz [3].

3.2 Odečítání tlaku z padů

Řešení je velmi podobné jako při odečítání napětí ze snímačů. Jediným rozdílem je použití padu místo struny. Ten rovněž generuje podle dynamiky hry různé napětí, které bychom museli pomocí A/D převodníku převést. Výhodou je absence strun a zamezení přeslechů. Možnou nevýhodou by mohl být tvar plochy, do které by se musel hráč trefovat. Příklady padů pro bicí nástroje viz [4].

3.3 Přerušení laserového paprsku

Tato varianta je založena na generování laserového paprsku namířeného na optický snímač. Přerušení paprsku se vyhodnotí jako zahrání noty. Tento způsob však neumožňuje snímat dynamiku, protože zde máme pouze dva stavy. Buď je paprsek přerušen, nebo nepřerušen. Na druhou stranu díky tomuto nepotřebujeme analogové vstupy s A/D převodem a můžeme využít přímo digitální vstupy. Možnou nevýhodou z pohledu hráče je neviditelnost laserového paprsku, který by musel být například pod umělou mlhou, aby šel vidět. Tento princip snímání je využit pro nástroj Laserová harfa [5].

3.4 Snímání na principu spínače

Zde je využit princip sepnutí a rozepnutí tlačítka. Každý tón má své tlačítko, které je v klidovém stavu rozepnuté. Když chce hráč zahrát tón, udeří do prvku, který tlačítko sepne. Volba tohoto prvku záleží na konstruktérovi. S použitím jednoho tlačítka tedy můžeme určit, kdy byla nota aktivována, zatím však máme málo informací na to, abychom dokázali určit dynamiku. To vyřešíme přidáním dalšího tlačítka, které umístíme tak, aby bylo tím samým prvkem stlačeno po stlačení prvního tlačítka. Z časového rozdílu vypočítáme sílu úderu. Čím větší je časová prodleva, tím slabší úder. Naopak čím kratší prodleva, tím je úder silnější. Na každý tón tedy potřebujeme dvě tlačítka a rovněž dva digitální vstupy. Tuto techniku snímání využívají digitální klávesové nástroje s rychlostním snímačem [6].

4 PROTOKOL MIDI

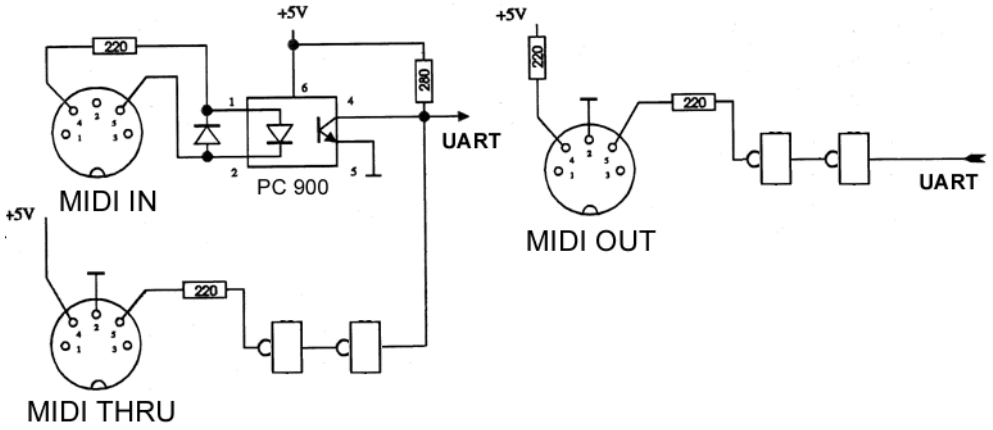
Název vznikl z počátečních písmen Musical Instrument Digital Interface, z čehož vyplývá, že jde o digitální rozhraní hudebního nástroje. Nápad vytvořit toto rozhraní přišel již v roce 1981 [6] a to především z důvodu potřeby komunikace mezi několika digitálními nástroji. Předtím byly syntezátory obrovské stroje, a to jak monofonní, tak později i polyfonní. Mezi tehdejší výrobce patřil ARP, Fairlight, Moog, Oberheim, Roland, Sekvential Circuits, Synclavier a Yamaha [7]. Když chtěl hráč docílit kombinaci zvuku syntezátorů různých výrobců, nezbývalo mu, než hrát tu samou melodii levou rukou na jeden syntezátor a pravou rukou na druhý. Propojení bylo možné jen mezi nástroji jednoho výrobce, které bylo navíc chráněno patenty, tudíž jej jiné firmy nemohly použít [7]. Tento nedostatek vyřešilo zavedení MIDI v roce 1983. Podrobná norma však vyšla až v roce 1985 [6].

4.1 Fyzická vrstva

Rozhraní MIDI používá 5-pinové konektory DIN označené většinou In, Out a Thru. In je vstupní konektor, Out výstupní a na konektor Thru jsou kopírována data z konektoru In. Konektory však nemusí být nutně tři. Některé přístroje mají pouze konektory In a Out, přičemž na výstup slučují interní a vstupní data. Této funkci se říká Soft Thru. Naopak některé přístroje mohou mít konektorů více, jak je ukázáno na obrázku níže.

Obr. 4.1 Příklad uspořádání MIDI konektorů [8]

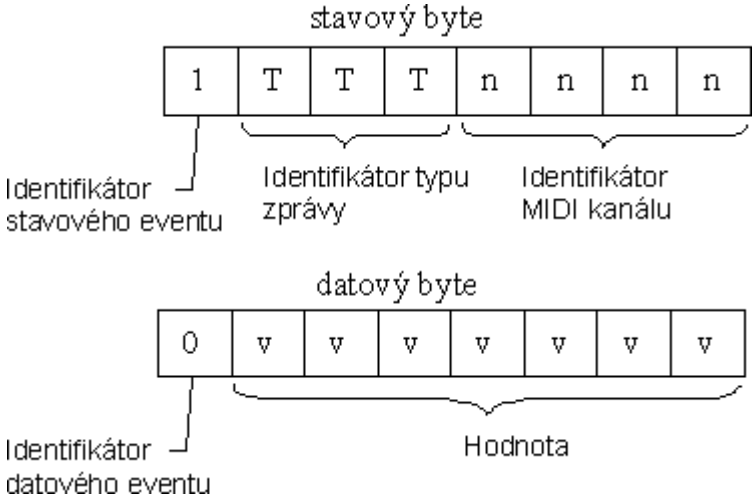
Samotná MIDI sběrnice je proudová smyčka. Logické nule odpovídá protékající proud 5 mA, při logické jedničce proud neprotéká. K přenosu dat je použit sériový asynchronní přenos rychlostí 31,25 kBaudů, délka trvání jednoho bitu je tedy 32 us. Rámec obsahuje jeden start bit, 8 datových bitů a dva stop bity.
Abychom se vyhnuli zemním smyčkám při spojení dvou přístrojů, nebývá zem na MIDI konektorech spojována se zemí přístroje a na vstupu bývá použit optoizolátor, který galvanicky propojené přístroje odděluje. Optoizolátor ovšem vnáší zpoždění, které by nemělo být větší než 2 ms. Z tohoto důvodu se nedoporučuje zapojovat přes Thru více než 4 zařízení za sebou [6].



Obr. 4.2 Schéma rozhraní MIDI [6]

4.2 Datový rámec MIDI

Základním nositelem informace je MIDI zpráva, která se skládá ze stavového bytu a podle potřeby několika datových bytů. Tyto byty bývají také označovány jako MIDI událost (MIDI event). Jeden MIDI event se skládá z celkem osmi bitů. První bit určuje, zda se jedná o stavový byte, tehdy je nejvýznamnější bit nastaven na 1, nebo datový byte, kdy je nejvýznamnější bit nastaven na 0. Díky tomuto jedinému bitu poznáme, zda byte nese stavovou informaci nebo data.



Obr. 4.3 Struktura stavového a datového bytu MIDI zprávy [6]

MIDI zprávy můžeme rozdělit podle toho, zda se váží k určitému kanálu, tehdy mluvíme o kanálových datech, nebo platí pro všechny kanály současně, těm říkáme systémová data. V případě kanálových dat slouží k identifikaci virtuálního kanálu spodní čtyři bity ve stavovém bytu. To nám dává celkem 16 možných kanálů, které můžeme využít například pro různé rejstříky.
Zbývající tři bity ve stavovém bytu slouží k identifikaci typu zprávy, na které se dále zaměříme podrobněji. Podle typu zprávy očekáváme následně několik datových bytů, které mohou přenášet hodnotu 0 až 127, protože máme k dispozici sedm bitů [6].
Pokud vysíláme systémová data, jsou spodní čtyři bity využity pro identifikaci typu systémových dat. Pomocí systémových dat lze přenášet informaci o čase, která slouží pro vzájemnou synchronizaci více zařízení, nebo informaci o tempu.

4.3 Typy kanálových MIDI zpráv

K určení typu zprávy slouží tři bity ve stavovém bytu. To nám dává na výběr osm druhů zpráv. Sedm z nich slouží pro kanálová data a jedna je určena pro identifikaci, že se jedná o systémová data [6, 7].

4.3.1 Nota zapnuta (Note on)

Identifikátorem je 1. Tato zpráva oznamuje aktivaci neboli zapnutí noty. Dále je třeba přenést o jakou notu se jedná. K tomu slouží první datový byte s rozsahem 0 až 127, čemuž odpovídají noty od sub-kontra C po šesti čárkované g6. Další datový byte nese informaci o dynamice ve shodném rozsahu. Nule odpovídá nota vypnuta, což má v praxi stejný význam jako bychom vysílali zprávu Note Off. Jednička znamená pianissimo piano, tedy co nejtišeji, a nejvyšší hodnota odpovídá fortissimo forte, tedy co nejhlasitěji. Jsme tedy schopni přenést celý dynamický rozsah. Celá MIDI zpráva se skládá z jednoho stavového a dvou datových bytů.

4.3.2 Nota vypnuta (Note off)

Identifikátorem je 0. Tato zpráva oznamuje vypnutí noty, což může znamenat například uvolnění klávesy. MIDI zpráva opět obsahuje datový byte nesoucí informaci o výšce noty a byte nesoucí dynamiku, s jakou byla nota uvolněna. Forma je tedy prakticky shodná s Note On.

4.3.3 Změna kontroleru (Control Change)

Identifikátorem je 3. Zpráva obsahuje dva datové byty z nichž v prvním je přenášeno číslo kontroleru a ve druhém je přenášena jeho nová hodnota. Posíláním těchto zpráv můžeme měnit parametry MIDI nástroje jako je hlasitost, modulace, volba režimů, informace o stavu pedálu, povely pro vypnutí všech not nebo zvuků a mnoho dalších parametrů.

4.3.4 Další typy MIDI zpráv

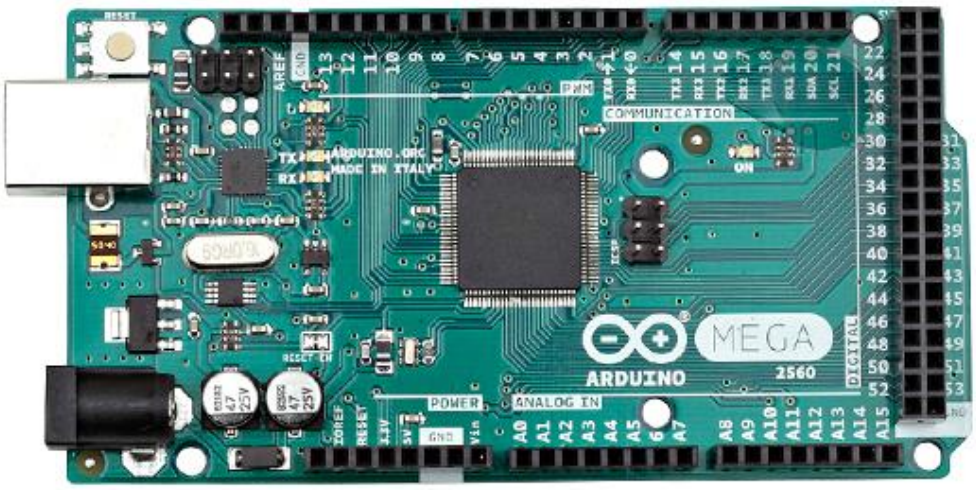
Kromě informací o notách můžeme přenášet další informace jako je individuální tlaková citlivost (ID = 2), změna programu (ID = 4), společná tlaková citlivost (ID = 5) a ohýbání tónu (ID = 6) [6]. Tyto zprávy pro nás však nejsou důležité, proto se jimi nebudeme zabývat podrobně.

5 VÝBĚR JEDNOČIPOVÉHO PROCESORU

Hlavním kritériem pro výběr jednočipového procesoru je dostatečný počet vstupů pro připojení všech snímačů a dále přítomnost alespoň jednoho sériového portu, který bude sloužit pro vysílání MIDI dat. Procesor musí být schopen data vysílat stanovenou rychlostí 31 250 bitů za sekundu. Těmto parametrům vyhovuje čip Atmel ATmega 2560 zakomponovaný ve vývojové desce Arduino MEGA 2560.

5.1 Popis vývojové desky Arduino MEGA 2560

Vývojová deska obsahuje jednočip, jehož vstupy a výstupy jsou vyvedeny na konektory typu pin. Dále je na desce umístěn stabilizátor napětí pro napájení čipu a dalších komponent. Potom krystal s frekvencí 16 MHz, který určuje taktovací frekvenci čipu. A převodník sériového portu na USB, sloužící k programování čipu. [9]



Obr. 5.1 Vývojová deska Arduino MEGA 2560 [9]

5.2 Popis čipu Atmel ATmega 2560

Tento jednočipový procesor nabízí 16 analogových vstupů a celkem 54 digitálních vstupně výstupních pinů. Zda se bude digitální pin chovat jako vstup nebo výstup se dá nastavit v software. Určené digitální piny také mohou sloužit jako čtyři sériové porty. Dále čip umožnuje vysílat pulzně modulovaný signál až na dvanácti pinech.
K programování se používá programovací jazyk Wiring, který se velmi podobá jazyku C. Díky přítomnosti USB převodníku stačí vývojovou desku připojit k PC přes USB kabel a nahrát program do čipu.

6 NÁVRH PROTOTYPU MIDI PŘEVODNÍKU PRO STUPNICI C DUR

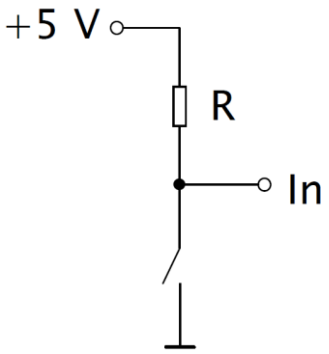
Na základě znalostí hudebního nástroje a různých typů snímačů jsme se s konstruktérem HW desky cimbalonu a s konstruktérem VST programu shodli využít snímání na principu tlačítek. Tento typ snímání umožnuje jednoznačné určení aktivity každé struny bez nežádoucích přeslechů a také určení dynamiky, s jakou hráč danou strunu vybudil. Pro každý tón použijeme dvě tlačítka, jejichž vhodné umístění zajistí konstruktér nástroje. Všechny tlačítka budou mít společnou zem, která se při stisknutí přivede na odpovídající digitální vstupní pin procesoru. Na každé tlačítko připadá jeden digitální vstup, což v případě zvoleného jednočipu dává možnost připojit až 53 tlačítek. V průběhu programu budeme kontrolovat jednotlivé vstupy a zjišťovat, zda je na ně přivedená logická úroveň LOW, kterou vstup aktivujeme. V případě, že vstup aktivován není, je na něj přes rezistor přivedeno kladné napětí +5 V, tedy logická úroveň HIGH.

Pro celý hudební nástroj toto zapojení však nebude stačit, protože potřebujeme minimálně 116 tlačítek, abychom pokryli všechny struny. Tento nedostatek vstupů můžeme jednoduše vyřešit pomocí zapojení do mřížky. Celkový počet tlačítek si rozdělíme na dva nebo více okruhů tak, abychom se vešli do počtu vstupů. Zvolme tedy tři okruhy. V tomto zapojení budou na jeden vstup připojeny tři tlačítka. Označme si je a, b, c. Ve výsledku budeme mít tlačítka 1a, 1b, 1c, 2a, 2b, 2c, 3a až po tlačítko 39b. Díky tomu obsadíme jen 39 vstupů. Zem, která byla původně společná pro všechna tlačítka, rozdělíme podle písmen na okruh-a, okruh-b a okruh-c, které rovněž připojíme na digitální piny. Ty nastavíme jako výstupy. Ve výchozím nastavení bude výstup okruh-a nastaven na logickou úroveň LOW, tedy zem. Ostatní budou nastaveny na logickou úroveň HIGH, tedy +5 V. Vstup reaguje na logickou úroveň LOW, takže se zmáčknutí tlačítka projeví jen na tlačítkách označených písmenem a.

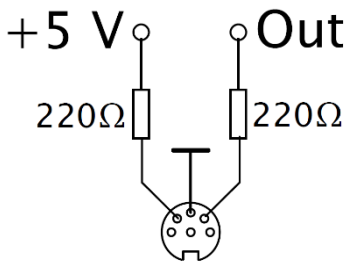
Program bude postupně kontrolovat aktivitu všech 39 vstupů, poté nastaví logickou úroveň LOW na okruh-b, na ostatní okruhy nastaví opačnou logickou úroveň, tedy HIGH. Až program opět projde všech 39 vstupů, aktivuje poslední okruh. Ve výsledku jsme schopni monitorovat 116 tlačítek na 39 vstupních pinech a 3 výstupních pinech. V případě požadavku většího počtu tlačítek můžeme mřížku rozšířit na čtverec, kdy dostáváme maximální využitelnost pinů. Pro velikost mřížky 26 x 26 dostaneme 676 tlačítek na pouhých 52 pinech.

Použití mřížky však s sebou nese jedno riziko. Na jeden vstup je připojeno více tlačítek. Jedno z nich má druhý kontakt připojený k zemi, ostatní na +5 V. Může zde nastat případ současného stlačení těchto dvou tlačítek, což má za následek

přímého spojení napájecího napětí a země, a tím vznik zkratu. Abychom tomuto spojení předešli, vřadíme mezi tlačítka diody, které omezí tok proudu na jeden směr. Zde se také nabízejí dvě možnosti umístění diod. První z nich je zařadit diodu mezi vstupní pin a tlačítko, druhou možností je zařadit diodu za výstupní pin, který slouží k aktivování okruhu. Z hlediska funkce diody je nepodstatné, kterou z těchto dvou variant použijeme. Z konstrukčního hlediska je však vhodnější použít způsob, který vykazuje nižší počet potřebných diod. Pro vzorový příklad se třemi okruhy a 116-ti tlačítky je vhodnější použít 3 diody, připojené k výstupním pinům.
Níže je uvedeno schéma zapojení převodníku. Na digitální vstup je připojen kontakt tlačítka, jehož druhý kontakt je spojen se zemí. Rezistor R je umístěn interně na čipu a není třeba jej připojovat externě. V klidovém stavu je tedy na digitálním pinu logická úroveň 1. Sériový výstup čipu je přiveden přes rezistor 220 Ω na pin č. 5 MIDI konektoru. Jelikož se jedná o proudovou smyčku, je pin č. 4 připojen přes rezistor k napájecímu napětí. Stínění kabelu je spojeno s prostředním pinem č. 2 a se zemí čipu.



Obr. 6.1 Principiální schéma zapojení tlačítka na vstup čipu



Obr. 6.2 Schéma zapojení výstupu čipu na MIDI konektor

6.1 Návrh programu

Jednočip je schopen provádět pouze jednu instrukci v jednom časovém okamžiku. Tyto instrukce ale provádí velmi rychle za sebou. V použití pro MIDI převodník potřebujeme neustále kontrolovat aktivitu na vstupních pinech, které odpovídají prvním tlačítkům daného tónu, budeme je nazývat primární. Pokud není aktivováno žádné tlačítko, nejsou na MIDI port vyslána žádná data. Když dojde ke stlačení libovolného primárního tlačítka musí program přestat kontrolovat ostatní primární tlačítka a zaměřit se na sekundární tlačítko tónu, který byl aktivován primárním tlačítkem. V tuto chvíli tedy program neustále kontroluje aktivitu na jediném tlačítku. Když zaznamená zmáčknutí sekundárního tlačítka, vyhodnotí časový rozdíl mezi těmito dvěma tlačítky. Na základě experimentu musíme zjistit, jakému časovému rozdílu odpovídá nejsilnější úder, který jsme schopni zahrát, a také nejslabší úder. Na tento rozsah nastavíme celkovou dynamiku. V čipu musíme dále přepočítat časový rozdíl na rozsah hodnot od 1 do 127. Nejmenší časový rozdíl a zároveň nejsilnější dynamika bude odpovídat hodnotě 127. Největší časový rozdíl, a tedy nejslabší dynamiku převedeme na hodnotu 1.

Může nastat situace, že z nějakého důvodu nebude sekundární tlačítko stisknuto. To by znamenalo přerušení chodu programu, a tedy i krátkodobou nefunkčnost převodníku, dokud by příslušné tlačítko nebylo stisknuto. Tomuto jevu můžeme zabránit tak, že budeme na stisk tlačítka čekat jen určený čas, který je potřebný k zahrání nejslabší dynamiky. Po uplynutí tohoto času se program opět vrátí ke kontrole primárních tlačítek.

Pokud program dospěje do bodu vyhodnocení dynamiky, máme vše potřebné k odeslání MIDI zprávy. Víme, o který tón se jedná, protože známe konkrétní tlačítko a známe také spočítanou dynamiku. Program tedy vyšle na sériový port stavový byte MIDI zprávy Nota zapnuta, následovaný dvěma datovými byty, první s informací o výšce noty a druhý s informací o její dynamice.

U cimbálu však neurčujeme délku trvání tónu délkou stisku klávesy, jak je tomu například u klavíru. K dispozici máme pouze impuls, kdy palička narazí na strunu. To, jak dlouho bude tón znít, určujeme pedálem. Pokud je pedál uvolněn, ozve se krátký zvuk. Pokud je sešlápnut, jsou dusítka zvednuta a tón se ozývá, dokud nedojde k uvolnění pedálu nebo struně nedojde energie. Délku tónu tedy určujeme společně pro všechny struny. Informaci o stavu pedálu vysílá - k tomu určená - MIDI zpráva. Aby byl tento nástroj kompatibilní i s jinými virtuálními nástroji, budeme po MIDI zprávě Note on - vysílat i zprávu Note off. Díky tomuto můžeme cimbálem ovládat například virtuální klavír. Pokud bychom zprávu Note off nevysílali, tóny by samy od sebe neutichly a ovládání jiných nástrojů mimo cimbálu by bylo velmi omezené.

6.2 Popis programu

Hlavní program se skládá ze tří základních částí. První částí je **pomocný podprogram s názvem nota**, který na základě vstupních informací vyšle MIDI zprávu. Těmito informacemi jsou výška tónu a dynamika. Na prvním řádku podprogramu je příkaz k vyslání stavového bytu zprávy Nota zapnuta. Pro veškerou komunikaci je použit první virtuální MIDI kanál. Na druhém řádku se nachází příkaz k vyslání prvního datového bytu s informací o výšce tónu a na třetím řádku je příkaz k vyslání druhého datového bytu s informací o dynamice. Vysílání MIDI zpráv probíhá přes sériový port. Každý příkaz v tomto případě vyšle jeden byte, tedy osm bitů, které jsou definovány MIDI standardem. Sériový port je nastaven tak, aby sám vysílal start bity i stop bity, což programátorovi ulehčuje práci, protože se zabývá vysíláním pouze osmi datových bitů.
Jako další příkaz následuje vyslání stavového bytu zprávy Nota vypnuta, který se provede se zpožděním 10 mikrosekund, aby zpráva o vypnutí noty nepřišla hned po zprávě zapnutí noty. Dalšími příkazy jsou opět datové byty s informací o výšce tónu, abychom vypnuli ten samý tón, který jsme před chvílí zapnuli, a informace o dynamice vypnutí tónu, která v případě použití pro elektronický cimbál není podstatná, ale vysíláme ji z důvodu kompletnosti zprávy.
Dalším blokem hlavního programu **je funkce setup**. Je to speciální blok programu, který slouží k nastavení čipu. Instrukce v něm se provedou jen jednou, a to po připojení napájení čipu nebo po jeho restartování. K restartování čipu slouží tlačítko na desce, umístěné vedle USB konektoru. Po jeho stisknutí program ukončí právě prováděnou činnost a začne se vykonávat od začátku, tedy od stejného místa jako po připojení napájení.
V tomto bloku se digitálním pinům přiřadí jejich funkce. Ty jsou celkem tři. Buď se digitální pin bude chovat jako výstup (OUTPUT), bude tedy vysílat data, nebo se nastaví jako vstup (INPUT) a bude data přijímat. Třetí možností je nastavení podobné vstupu, kde je navíc interně vřazen rezistor (INPUT PULLUP), a to mezi digitální pin a napájecí napětí +5 V. Výchozí hodnotou je tedy logická úroveň HIGH, protože je na vstupu přítomno kladné napětí. Abychom vstup aktivovali, musíme pin uzemnit. Tehdy bude na pinu 0 V a přes rezistor bude protékat proud. Protože rezistor bývá v řádu desítek až stovek kΩ, protéká rezistorem zanedbatelný proud, menší než 1 mA. Toto zapojení se používá k jednoznačnému rozpoznání digitální úrovně. V případě, že by rezistor nebyl zapojen a na vstup by nebylo přivedeno žádné napětí, mohlo by dojít k indukci napětí na vodiči připojenému k pinu, nebo k jiným nežádoucím jevům. Použitím rezistoru zajistíme, že bude na vstupu vždy jen +5 V nebo 0 V.

V našem případě nastavíme vstupní piny 22 až 37 na režim vstupu s připojeným rezistorem. Využijeme tedy 16 vstupů, ve kterých pokryjeme stupnici C dur s možností snímání dynamiky úhozu. Abychom nemuseli psát 16 podobných příkazů pro nastavení všech vstupů, je vhodné využít cyklus for, u kterého je předem daný počet opakování. Cyklus projde piny 22 až 37 a nastaví na nich režim INPUT PULLUP.

Dalším krokem ve funkci setup je zahájení sériové komunikace na sériovém portu číslo 3, rychlostí 31 250 bitů za sekundu, což je rychlost rovněž specifikována normou MIDI. Tento příkaz musíme použít, abychom mohli na sériový port vysílat data.

Posledním článkem hlavního programu **je funkce loop**. Je to smyčka, která se provádí s nekonečným počtem opakování poté, co program prošel funkcí setup. V této smyčce kontrolujeme stav na digitálních vstupech. Protože program nedokáže zpracovávat více instrukcí najednou, kontroluje jako první vstup 22, pokud vstup není aktivován, pokračuje dále a kontroluje vstup 24, pokud tento vstup rovněž není aktivován, pokračuje na další. Až program prověří všechny vstupy, vrátí se zpět na začátek smyčky loop. K zjišťování stavu na vstupu je použita funkce if, která provede pomocnou funkci, pokud platí její podmínka. V našem případě je podmínkou, je-li na digitálním pinu přítomno napětí 0 V, což odpovídá logické úrovni 0. Vstupy jsou rozvrženy tak, že každý sudý vstup n odpovídá prvnímu tlačítku tónu, lichý vstup n+1 slouží k rozpoznání dynamiky úhozu. Pokud není sepnuto žádné tlačítko, prochází program pouze sudé vstupy.

Aby se zabránilo opakovanému odesílání MIDI zpráv při spojitém držení tlačítek, byla do kódu přidána následující podmínka. Program vyhodnotí aktivitu tónu, pouze když je zmáčknuto sudé tlačítko a zároveň není stlačeno následující liché tlačítko. Čip pracuje tak rychle, že než dojde k uvolnění tlačítka, může být jeho aktivita již několikrát otestována.

V případě, kdy je na sudém pinu logická úroveň 0, přestává program kontrolovat ostatní sudé vstupy a uloží si do pomocné proměnné označené *cas* údaj o čase. To bude naše výchozí hodnota pro počítání dynamiky. Nyní program kontroluje aktivitu druhého, tedy lichého tlačítka. Pro případ, kdy by z jakéhokoli důvodu nedošlo ke stisknutí tlačítka, kontroluje program aktivitu na vstupu jen po dobu 100ms. Pokud aktivitu nezaznamená, vrátí se ke kontrole primárních tlačítek tónů. V okamžiku, kdy dojde ke stisknutí i druhého tlačítka, zapíše se do proměnné s názvem *cascelkem* hodnota rozdílu aktuálního času a času uloženého při stisku prvního tlačítka. Protože potřebujeme mít údaj o dynamice v daném rozsahu 1 až 127, je potřeba údaj o čase přizpůsobit. Jako první je potřeba provést měření, jakému rozdílu času odpovídá nejsilnější a nejslabší úder, na základě toho poté stanovíme dynamický rozsah, do kterého pomocí funkce map časové hodnoty

převedeme. Funkce map má celkem pět parametrů. Zadáváme do nich číslo nebo proměnnou, kterou chceme převést do jiného rozsahu. Dalšími parametry jsou minimum a maximum rozsahu zadávaného čísla, a minimum a maximum rozsahu, do kterého chceme číslo úměrně převést. Funkce tedy změní měřítko. Správným nastavením rozsahu dostaneme z delšího času nižší číslo a z kratšího času větší číslo. Jako prozatímní hodnoty pro zkušební účely je použit rozsah 0 ms až 100 ms. Pro finální výrobek bude potřeba tyto časy změřit, jsou totiž závislé na fyzické vzdálenosti snímačů a také dynamiky, která bude použitá mechanika dosahovat. Na základě výsledků měření stanovíme, zda bude vhodnější časový rozdíl měřit v milisekundách, nebo v mikrosekundách. Tomuto bychom také měli přizpůsobit dobu čekání na druhé tlačítko, která by měla být těsně za hranicí času potřebného pro zahrání nejslabšího úderu. K čekání používáme podmínku while, která provádí cyklus, dokud platí podmínka. V našem případě, pokud rozdíl časů není větší než 100 ms.

Teď, když máme informaci o dynamice ve správném rozsahu, zbývá poslat MIDI zprávu Nota zapnuta. K tomu slouží předem vytvořená funkce s názvem **nota**, zmiňovaná na začátku. Jejími parametry jsou výška tónu a dynamika. Co se týče zadávaných hodnot, můžeme využívat buď desítkovou soustavu nebo hexadecimální soustavu. Zde je výhodné, že čtyřem znakům v binární soustavě odpovídá jeden znak v hexadecimální soustavě. Čísla se tedy dají převádět bez použití kalkulačky. K identifikaci datového bytu slouží nejvýznamnější bit, který je nastaven na 0. Díky tomu můžeme poslat jen číslo noty nebo dynamiky bez dalších úprav. Po provedení podprogramu nota se program vrátí ke kontrole následujících primárních tlačítek a případně dalším vysíláním MIDI zpráv.

7 PRAKTICKÝ NÁVRH MIDI PŘEVODNÍKU PRO CIMBÁL

V předchozí kapitole byl podrobně popsán způsob provedení převodníku pro stupnici C dur (c1 – c2) s použitím 16-ti tlačítek na principu rychlostního snímání dynamiky úhozu. Jelikož na projektu elektronického cimbálu souběžně pracujeme tři a každý se zabývá různou částí nástroje, začal jsem svou část, tedy MIDI převodník, řešit tímto způsobem. Struny jsem tedy pro zkušební účely nahradil tlačítky a výstupní konektor jsem zapojil do MIDI vstupu kláves, které MIDI zprávy přehrávaly. To mi dávalo možnost ověřit si funkčnost převodníku. Po několika úpravách programu jsem dosáhl funkčnosti a stability prototypu převodníku, ze kterého budu vycházet a rozšířím jej pro potřeby cimbálu.
Oproti prvotnímu způsobu řešení snímání strun, tedy použití dvou tlačítek, došlo k velké změně. Konstruktér nástroje zvažoval různé způsoby uchycení tlačítek, nebo jiných snímačů na principu spínání kontaktu, až došel k závěru, že nahradí primární snímač uzavřenou gumovou trubicí, na jejímž konci bude elektretový mikrofon. Druhý konec trubice bude utěsněn. Z hlediska hry na nástroj má toto řešení výhodu, kdy nezáleží, v jaké části je na strunu zahráno, protože trubice je po celé délce struny, kdežto tlačítko pokrývá pouze její část. Signál z elektretového mikrofonu je však analogový a má malou úroveň napětí. Proto k němu konstruktér sestavil obvod, který funguje jako trigger, neboli spouštěč. Tento obvod má nastavitelnou citlivost, takže je možnost jej doladit pro každou strunu individuálně. Podle informací konstruktéra by výstupem obvodu v klidovém stavu mělo být napětí 0 V, v případě úderu pak 3 V. Jak jsme si však výstupní napětí pomocí vývojové desky Arduino zobrazili v grafu, neodpovídalo úplně ideálnímu stavu. Očekával jsem průběh s minimální nástupnou a sestupnou hranou a zároveň co nejkratší, nejlépe v řádu milisekund. Sestupná hrana však trvala téměř jednu sekundu, což je pro zpracování signálu procesorem nevhodné, protože při zapojení do digitálního vstupu je úroveň logické hodnoty HIGH vyhodnocena pro napětí 3 až 5 V a logická úroveň LOW pro napětí 0 až 2 V [9]. Oblasti, která se nachází mezi 2 až 3 V se říká oblast hystereze. Je zde proto, aby nedocházelo k rychlým změnám logické úrovně, když se vstupní napětí pohybuje kolem mezní úrovně. Obvod dává při úderu na strunu kolem 4 V, oproti katalogovým 3 V, což je v našem případě spíše výhoda než problém. Ovšem doba, než napětí klesne ze 4 V na 2 V, je až příliš dlouhá a nelze tedy vyhodnotit dva rychlé po sobě jdoucí údery. Než stihne napětí klesnout pod 2 V jsou na výstupu opět 4 V a digitální vstup se tedy mezitím nedostal do stavu logické úrovně LOW. Tyto údery tedy splynou do jednoho.

Problém by šel vyřešit využitím analogových vstupů čipu. V programu by se následně signál upravil tak aby byla brána pouze špička, tedy například napětí nad 3,7 V. Takto upravený signál již bude mnohem kratší a lépe by se tedy zpracovávaly reakce na údery. Ovšem čip disponuje pouze 16-ti analogovými vstupy a zároveň v tomto případě není možné využít zapojení do mřížky, protože výstupem obvodu je napětí, na rozdíl od tlačítka, které pouze spíná kontakt. Po konzultaci s konstruktérem jsme se tedy dohodli, že úpravu signálu provedeme ještě před vstupem do čipu, a to s použitím komparátoru.

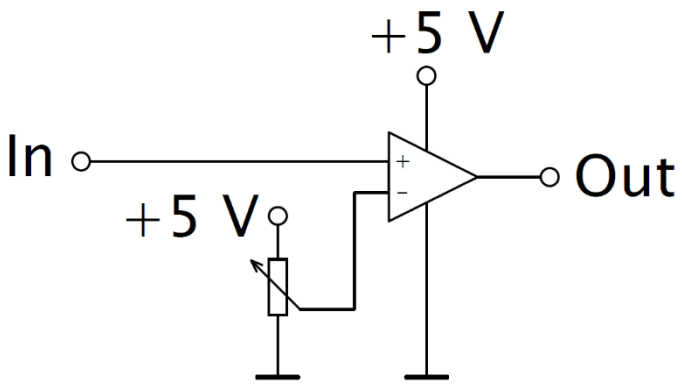
7.1 Popis funkce a zapojení komparátoru

Komparátor je obvodové zapojení, které ke své funkci využívá operační zesilovač. Může jít také o specializovaný obvod určený do analogově digitálních převodníků, kde je vyžadována velmi vysoká rychlost komparace [10]. Pro naše potřeby si ale vystačíme s operačním zesilovačem. Jak již název napovídá, dochází zde k porovnávání napětí. Operační zesilovač má dva vstupy, jeden invertující (-), druhý neinvertující (+), a výstup. Jde o zesilovač, který zesiluje rozdíl těchto dvou vstupů. Jeho charakteristickými vlastnostmi jsou velký vstupní odpor a malý výstupní odpor. Slouží tedy zároveň jako oddělovací člen, protože jeho vstupem nezatěžujeme výstup předchozího obvodu. Další zásadní vlastností, důležitou pro funkci komparátoru, je jeho velké zesílení, které bývá v řádu 100 000 [10]. Pokud bychom toto zesílení chtěli zmenšit, použili bychom zápornou zpětnou vazbu. Komparátor však využívá plného zesílení operačního zesilovače, zpětná vazba tedy není zapojena.

Jak již bylo psáno, operační zesilovač zesiluje rozdíl vstupních napětí. Rozkmit výstupního napětí je však dán jeho napájecím napětím, které je v našem případě 5 V. V praxi je ale výstupní napětí nižší než napájecí, pokud nejde o operační zesilovače typu rail-to-rail [11], které jsou schopny na výstup dodat téměř plný rozkmit napájecího napětí.

Když na neinvertující vstup (+) přivedeme napětí 4 V a na invertující vstup napětí 3,7 V, je jejich rozdíl násoben 100 000x. Což dává výsledek 30 000 V. Toto napětí je omezeno a na výstupu naměříme napětí 3,9 V. Dojde tedy k maximálnímu možnému vybuzení výstupu. Tomuto jevu se říká saturace. V opačném případě, je-li na invertujícím vstupu (-) větší napětí než na neinvertujícím (+), bude na výstupu 0 V. Výsledek by totiž byl záporný, ale protože je záporná svorka napájecího napětí připojena na zem (0 V), nemůže být na výstupu napětí nižší než na této svorce.

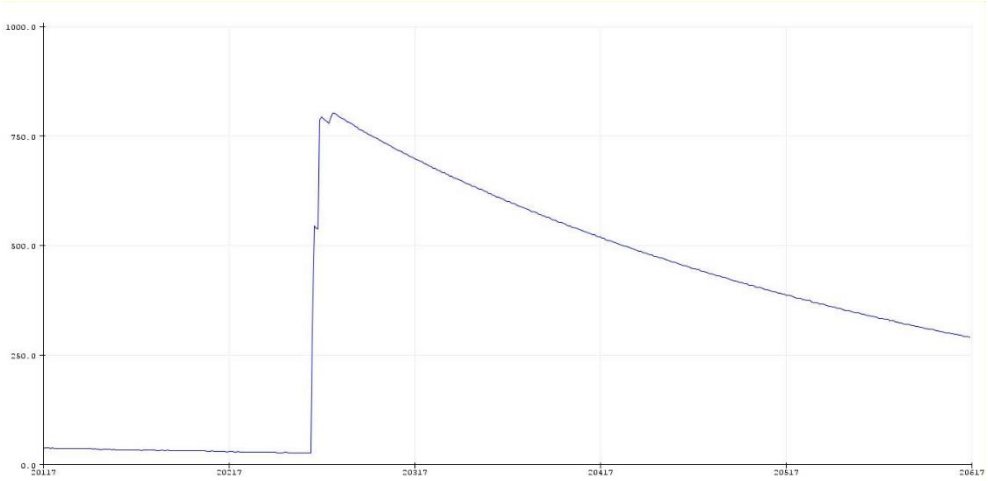
Pro náš případ bude vhodné porovnávat napětí z triggeru s referenčním napětím. Referenční napětí získáme z napájecího rozvodu 5 V. Jelikož je celkové napájecí napětí stabilizováno, není nutné řešit samostatnou stabilizaci. Stačí tedy navrhnout dělič napětí, na jehož výstupu bude 3,7 V. Toto napětí musí být přivedeno na invertující vstup (-) všech komparátorů. Není nutné, aby měl každý komparátor svůj vlastní dělič napětí, proto bude stačit jeden centrální. Z důvodu variability a případné možnosti doladění, jsme se shodli, že místo dvou rezistorů, které by klasický dělič obsahoval, použijeme potenciometr o velikosti 10kΩ. Viz obr. 7.1.



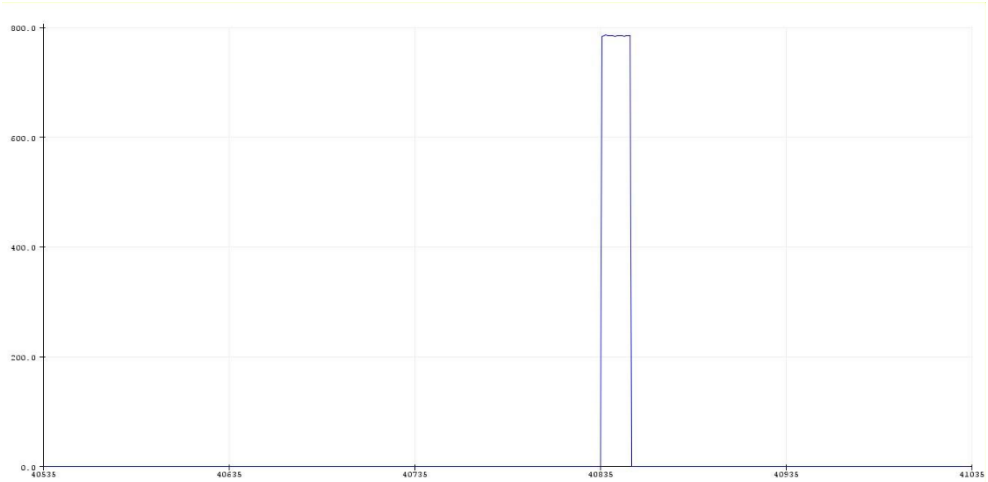
Obr. 7.1 Principiální schéma zapojení komparátoru s referenčním napětím

To nám v případě potřeby dovolí napětí doladit, což se projeví v prodloužení nebo zkrácení délky pulzu. Tento potenciometr by měl být ukryt v nástroji tak, aby se k němu hráč za normálních okolností nedostal a nedošlo tak v extrémním případě až k dočasné nefunkčnosti nástroje zapříčiněné nevhodně nastaveným komparačním napětím. Doporučuji začít s napětím nastaveným na 3,7 V. Pro vyšší hodnoty dostaneme sice kratší puls, ale nabízí se zde riziko, že se výstupní napětí jednotlivých triggerovacích obvodů bude lišit a nemuselo by tuto hodnotu překročit, takže by nedošlo k aktivaci vstupu. Pokud by se vyskytla potřeba referenční napětí dodatečně stabilizovat, můžeme využít jednoduchého zapojení pro stabilizaci s použitím Zenerovy diody [10].
Na obrázku 7.2 a 7.3 je vidět původní napětí z triggerovacího obvodu a poté napětí zpracované komparátorem s komparačním napětím 3,7 V. Tyto průběhy byly vyobrazeny přímo prostřednictvím vývojové desky Arduino pomocí analogového vstupu. Na ose X je pořadí přijaté hodnoty po sériovém kanále do PC a na ose Y se nachází rozsah napětí 0 až 5 V zakódovaný do deseti bitů, rozsah je tedy od 0 až do 1024. Přesný popis os zde však není úplně podstatný, jde především o tvar signálu.

Průběhy byly přes čip navzorkovány odděleně, protože program umožňuje vykreslit pouze jeden graf. Proto tedy nezačínají ve stejném bodě osy X. V ideálním případě by se měly nástupné hrany obou průběhů překrývat.



Obr. 7.2 Napětí na výstupu triggeru po vybuzení struny

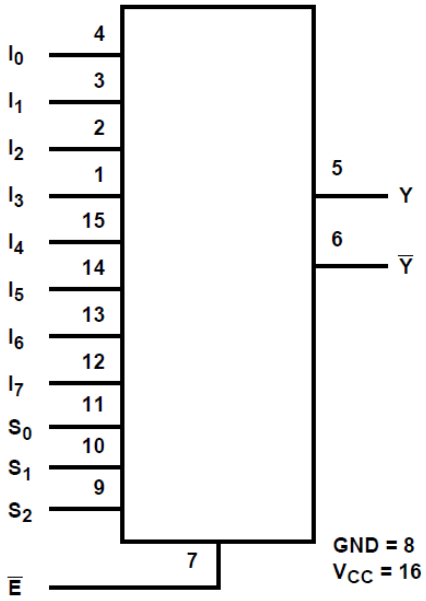


Obr. 7.3 Napětí z triggeru zpracované komparátorem

Teď když už má signál tvar pulzu, je jeho průběh podobný průběhu stisknutí tlačítka. Stále však nelze využít zapojení do mřížky, protože nemůžeme jednoduše spojit výstupy komparátorů. Navíc u komparátorů nelze řídit výstup ve smyslu zapnutí a vypnutí. Nemohli bychom je tedy aktivovat, aby se chovaly jako řádky v mřížce. Zde jsme se tedy rozhodli využít multiplexer.

7.2 Popis funkce a zapojení multiplexeru

Multiplexer je integrovaný obvod, který funguje jako elektronický přepínač. Vyrábí se spousta různých druhů, lišících se v počtu vstupů a počtu přepínačů v jednom pouzdře. Našim požadavkům vyhovoval multiplexer CD74HCT151. Jedná se o vysokorychlostní typ s osmi datovými vstupy, třemi adresovými vstupy a dvěma výstupy, z toho jedním negovaným. Negovaný výstup má stejnou funkci jako běžný výstup, jeho data jsou ale posílány v opačné logické úrovni. Popis všech pinů multiplexeru je na obrázku 7.4 kde I_x jsou datové vstupy, S_y adresové vstupy, E značí povolovací vstup enable a Y jsou výstupy z nichž výstupní pin 6 je negovaný.



Obr. 7.4 Popis vstupních, výstupních a napájecích pinů multiplexeru [12]

Tento multiplexer je digitální a pracuje tedy přímo s logickými úrovněmi. Používá se v případech, kdy je potřeba zvýšit počet vstupů. Principem multiplexeru je přepínání. Z osmi vstupů volíme jeden, který bude přiveden na výstup. Volba je provedena pomocí adresových vstupů. Ty jsou celkem tři, což dává osm kombinací. Proto tedy multiplexer obsahuje osm vstupů. Zjednodušená pravdivostní tabulka viz Tab. 7.1, kde L je logická úroveň LOW, H logická úroveň HIGH a X značí, že logická úroveň nemá vliv.

Tab. 7.1 Zjednodušená pravdivostní tabulka multiplexeru

Adresové vstupy			Datové vstupy								Výstup
S2	**S1**	**S0**	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**Y**
0	0	0	L	X	X	X	X	X	X	X	L
0	0	0	H	X	X	X	X	X	X	X	H
0	0	1	X	L	X	X	X	X	X	X	L
0	0	1	X	H	X	X	X	X	X	X	H
0	1	0	X	X	L	X	X	X	X	X	L
0	1	0	X	X	H	X	X	X	X	X	H
0	1	1	X	X	X	L	X	X	X	X	L
0	1	1	X	X	X	H	X	X	X	X	H
1	0	0	X	X	X	X	L	X	X	X	L
1	0	0	X	X	X	X	H	X	X	X	H
1	0	1	X	X	X	X	X	L	X	X	L
1	0	1	X	X	X	X	X	H	X	X	H
1	1	0	X	X	X	X	X	X	L	X	L
1	1	0	X	X	X	X	X	X	H	X	H
1	1	1	X	X	X	X	X	X	X	L	L
1	1	1	X	X	X	X	X	X	X	H	H

Posledním pinem multiplexeru je povolovací vstup enable. Jelikož je tento vstup negovaný, musíme na něj přivést zem, tedy 0 V, aby obvod fungoval. Bez přivedení země by obvod nebyl aktivní a na výstupu by se neobjevila žádná data.
Pro elektronický cimbál jsme se rozhodli využít pouze čtyři datové vstupy, které můžeme řídit dvěma adresovými vstupy. To především k ušetření času, který by se jinak spotřeboval vysíláním dat na adresové vstupy a neustálým přepínáním. Protože potřebujeme obstarat celkem 58 strun, použijeme 16 multiplexerů a tedy i 16 digitálních vstupů na čipu.

Tab. 7.2 Rozpis připojení primárních snímačů na vstupy multiplexerů

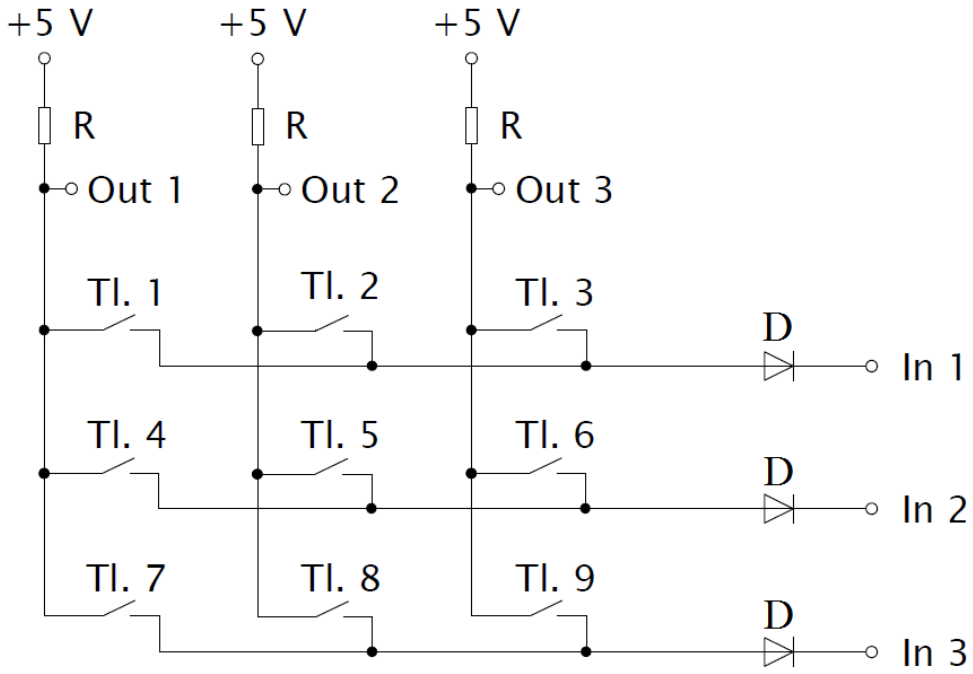
Číslo multiplexeru	Struna na vstupu 1	Struna na vstupu 2	Struna na vstupu 3	Struna na vstupu 4
1	C	e	gis_1	c_3
2	Cis	f	a_1	cis_3
3	D	fis	ais_1	d_3
4	Dis	g	h1	dis_3
5	E	gis	c_2	e_3
6	F	a	cis_2	f_3
7	Fis	ais	d_2	fis_3
8	G	h	dis_2	g_3
9	Gis	c_1	e_2	gis_3
10	A	cis_1	f_2	a_3
11	Ais	d_1	fis_2	-
12	H	dis_1	g_2	-
13	c	e_1	gis_2	-
14	cis	f_1	a_2	-
15	d	fis_1	ais_2	-
16	dis	g_1	h_2	-

Adresové vstupy budou sdílet stejná data, vysílaná rovněž z čipu. Po nastavení adresy zbývá přečíst data postupně od každého multiplexeru. Až přečteme všech šestnáct, změníme adresu a můžeme číst dalších šestnáct a tak dále, než přečteme všech 58 strun.

I když si v tomto případě vystačíme pouze se dvěma adresovými vstupy, je nutné zbývající adresový vstup přivést na zem. Jinak by se obvod nemusel chovat správně, protože bychom nepřivedli jednoznačnou logickou úroveň. Pro tuto část celkového systému snímačů využijeme z čipu 16 vstupů a 2 výstupy, celkem tedy 18 pinů.

7.3 Zapojení sekundárních snímačů

Sekundární snímače, umístěné pod strunou, tvoří podle původního plánu tlačítka. Zde tedy využijeme zapojení do mřížky. Pod každou strunou se nachází dvě tlačítka. Jedno je umístěno uprostřed struny, druhé na kraji. Výjimku tvoří čtveřice nejkratších strun, které mají tlačítko jen jedno. Celkem tedy potřebujeme připojit 112 tlačítek. Z hlediska přehlednosti programu jsme se rozhodli použít mřížku o velikosti 16 řádků na 7 sloupců. Principiální schéma zapojení mřížky se třemi řádky a třemi sloupci je na obrázku 7.5. Pro naše potřeby je mřížka rozšířena o potřebný počet dalších řádků a sloupců, princip zapojení ale zůstává shodný.



Obr. 7.5 Principiální schéma zapojení mřížky

Toto řešení je výhodné zejména proto, že používáme také 16 multiplexerů. Záměrem je, aby na jeden okruh multiplexerů připadaly dva řádky mřížky. Program bude tedy přehlednější. Tím, že nejvyšší struny obsahují pouze jeden spínač, není nutné, aby měla mřížka osm sloupců, ale vystačí si s plně obsazenými sedmi. To opět šetří čas potřebný k přepínání sloupců.
Pořadí připojených snímačů je následující. Mřížka začíná prvním řádkem a prvním sloupcem. Zde je připojen prostřední snímač struny C. Jelikož aktivujeme řádek, leží následující snímače ve směru prvního řádku. Následuje krajní snímač tónu C,

poté prostřední snímač tónu Cis, krajní snímač tónu Cis a tak dále až do konce řádku, který končí krajním snímačem tónu G. Detailní pořadí je uvedeno v tabulce 7.3 kde *s* značí střední snímač, *k* značí krajní snímač, struny bez označení mají snímač jen jeden.

Tab. 7.3 Rozpis připojení sekundárních snímačů do mřížky

Sloupec mřížky	Řádek 1	Řádek 2	Řádek 3	Řádek 4	Řádek 5	Řádek 6	Řádek 7
1	C – s	Gis – s	e – s	c_1 – s	gis_1 – s	e_2 – s	c_3 – s
2	C – k	Gis – k	e – k	c_1 – k	gis_1 – k	e_2 – k	c_3 – k
3	Cis – s	A – s	f – s	cis_1 – s	a_1 – s	f_2 – s	cis_3 – s
4	Cis – k	A – k	f – k	cis_1 – k	a_1 – k	f_2 – k	cis_3 – k
5	D – s	Ais – s	fis – s	d_1 – s	ais_1 – s	fis_2 – s	d_3 – s
6	D – k	Ais – k	fis – k	d_1 – k	ais_1 – k	fis_2 – k	d_3 – k
7	Dis – s	H – s	g – s	dis_1 – s	h1 – s	g_2 – s	dis_3 – s
8	Dis – k	H – k	g – k	dis_1 – k	h1 – k	g_2 – k	dis_3 – k
9	E – s	c – s	gis – s	e_1 – s	c_2 – s	gis_2 – s	e_3 – s
10	E – k	c – k	gis – k	e_1 – k	c_2 – k	gis_2 – k	e_3 – k
11	F – s	cis – s	a – s	f_1 – s	cis_2 – s	a_2 – s	f_3 – s
12	F – k	cis – k	a – k	f_1 – k	cis_2 – k	a_2 – k	f_3 – k
13	Fis – s	d – s	ais – s	fis_1 – s	d_2 – s	ais_2 – s	fis_3
14	Fis – k	d – k	ais – k	fis_1 – k	d_2 – k	ais_2 – k	g_3
15	G – s	dis – s	h – s	g_1 – s	dis_2 – s	h_2 – s	gis_3
16	G – k	dis – k	h – k	g_1 – k	dis_2 – k	h_2 – k	a_3

Oproti původnímu plánu použít interní rezistor umístěný v čipu jsem se rozhodl využít klasický rezistor, o velikosti 10 kΩ, napájený centrálním rozvodem napájení. Je to z důvodu zkrácení cesty proudu, napájecí mřížku. Proud je sice v řádu mA, ale díky této změně nebude odebírán z čipu a po datových vodičích půjde jen logická úroveň napětí. Na konci řádku mřížky se nachází dioda, jejíž činnost byla popsána v předchozí kapitole.

7.4 Rozšíření programu pro potřeby cimbálu

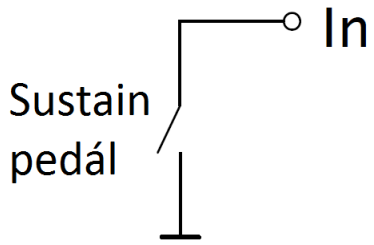
V každém bloku programu došlo k výrazným změnám. V podprogramu **setup** bylo nutné přidat další vstupy a výstupy. Ty jsem si rozdělil do pěti kategorií. Výstupy pro řízení adresy multiplexerů na pinech 1 a 2, vstupy pro data z multiplexerů na pinech 22 až 37, výstupy pro aktivování řádků mřížky na pinech 4 až 10, vstupy pro připojení sloupců mřížky na pinech 38 až 53 a opět sériový port pro vysílání MIDI zpráv na pinu 14. Piny jsem rozdělil smysluplně tak, aby se nacházely na multipinovém konektoru desky Arduina vedle sebe podle způsobu použití.
Dále je nutno dodat, že jsem zde také nastavil výchozí hodnoty výstupů. Výstupy pro řízení adresy multiplexerů jsou nastaveny na logické úrovně LOW a výstupy pro aktivování řádků mřížky jsou nastaveny na logickou úroveň HIGH. Není tedy aktivován žádný řádek a na multiplexerech je naadresován první okruh. V případě, že bychom toto nastavení neprovedli, mohlo by v prvním projití funkce loop dojít k nesprávnému vyhodnocení místa úderu a zahrání tónu jiné struny, než byla vybuzena.
V podprogramu **loop** se rozšířil počet podmínek (if), potřebných ke zjišťování aktivity strun, z původních 8-mi tónů na 58. Tyto podmínky jsou rozděleny v blocích po osmi. Na začátku dojde k nastavení adresy multiplexerů. Tato adresa je společná pro dva bloky podmínek, protože máme 16 multiplexerů. Je nastaven první okruh, tedy prvních šestnáct nejnižších strun. Než budeme zjišťovat aktivitu strun, je také podstatné aktivovat první řádek mřížky. První řádek je připojen k pinu č. 4 a aktivujeme jej přivedením logické úrovně LOW. Před tímto příkazem je však zdánlivě zbytečný příkaz k deaktivování posledního řádku mřížky, to bude vysvětleno v průběhu. Teprve teď můžeme kontrolovat stav na vstupních pinech, odpovídajících prvním osmi strunám, než dojdeme na konec řádku mřížky. Následně musíme deaktivovat první řádek a aktivovat druhý. Protože jsme v podprogramu setup nastavili výchozí stav všech řádků, budeme předpokládat, že je vždy aktivní pouze jeden řádek, nemusíme tedy při každé změně nastavovat logickou úroveň na všech sedmi řádcích, ale pouze na dvou. Teď můžeme zjistit stav dalších osmi strun. Protože jsme došli k šestnácté struně, je třeba změnit adresu multiplexeru a aktivovat další řádek mřížky. Tímto způsobem dojdeme až k poslední struně. Podprogram loop má tu vlastnost, že se neustále opakuje. Na konci je však aktivovaný poslední řádek mřížky, který musíme na začátku deaktivovat. Proto tedy ta instrukce, která se zprvu zdála zbytečná.
Nyní se zaměříme na samostatné vyhodnocení aktivity struny. Oproti prvotnímu návrhu, kdy se aktivuje první primární snímač a poté sekundární, vnáší obvod pro zpracování signálu z elektretového mikrofonu, komparátor a multiplexer do cesty signálu primárního snímače určité zpoždění. To zapříčinilo, že signál

ze sekundárního snímače předbíhá signál z primárního. Řešením je změnit program tak, aby reagoval na sekundární snímač a podle primárního určil dynamiku úhozu. S tímto řešením nebylo zprvu počítáno, ale je způsobem, jak umožnit snímání dynamiky.
Co zůstalo zachováno z původního návrhu, je kontrola odeznění reakce na předchozí úder. Zde je dobré zmínit potenciometr k nastavení komparačního napětí, popisovaný v podkapitole 7.1. Jelikož se prohodilo pořadí snímačů, program nevyhodnotí nové zahrání struny, pokud je signál z mikrofonu na logické úrovni HIGH. Došlo by totiž k vyhodnocení chybné dynamiky, protože tento signál ještě neodezněl z předchozího vybuzení struny. Tím, že díky potenciometru máme možnost ovlivňovat délku trvání logické úrovně, můžeme trvání zkrátit na dobu nezbytně nutnou pro vyhodnocení aktivity, ale zároveň ne příliš, aby nebylo komparační napětí vyšší než napětí z jednotlivých triggerovacích obvodů. Zjednodušeně lze říci, že program vyhodnotí zahrání struny je-li aktivní první snímač ale druhý ne. V případě splnění této podmínky je volána funkce s názvem ton, které jako parametr posíláme číslo noty podle normy MIDI a číslo pinu, na který je připojen odpovídající druhý snímač pro určení dynamiky. Oproti původnímu návrhu, kdy byla část programu pro počítání dynamiky pro každou strunu zvlášť, došlo ke zjednodušení a zpřehlednění. Program je také méně náročnější na paměť a lépe se provádějí případné změny.
Nástroj je vybaven dvojicí spínačů pod každou strunou, u kterých bylo zamýšleno snímat více míst vybuzení struny, v našem případě uprostřed a na kraji, a následně přehrát odpovídající zvukový vzorek. Zvuk zahraný na kraji struny by měl obsahovat větší podíl harmonických frekvencí než zvuk zahraný uprostřed struny. Tímto by se dosáhlo většího dojmu z hry, protože by se nástroj více blížil reálnému nástroji a zároveň by nabízel větší možnosti hry. Od tohoto řešení se muselo upustit, protože podle informací vývojáře software, přehrávajícího zvukové vzorky, je již příliš vytížena operační paměť počítače (RAM), ve které musejí být zvuky načteny. Rozšíření zvukové banky o další zvukové vzorky by vedlo k potřebě navýšení operační paměti na dvojnásobek. Nicméně z konstrukčního hlediska je nástroj na toto řešení připraven.
Další částí celého programu je pomocná funkce **ton**. Jejími parametry jsou výška tónu a číslo pinu, podle kterého má počítat dynamiku. Princip počítání dynamiky zůstal z prvotního návrhu zachován. Na počátku je uložen čas běhu programu v mikrosekundách a při aktivitě vstupu zadaného parametrem je od aktuálního času uložený čas odečten a následně převeden do rozsahu 1 až 127. I když je rozsah MIDI normou stanovený od nuly, nemá smysl s nulou počítat, protože při poslání má ten samý význam jako nota vypnuta.

Do cyklu *while* byla také přidána další podmínka. Vytvořil jsem si pomocnou proměnnou, která je nastavena na logickou pravdu (true). Poté, co je vyslána MIDI zpráva, je tato proměnná přepsána na logickou nepravdu (false). Proměnnou jsem zakomponoval do podmínky tak, aby cyklus nezabíral zbytečný strojový čas. Před touto úpravou byla délka cyklu pevně dána a odpovídala času potřebnému pro nejnižší dynamiku. Nyní tato délka odpovídá zahrané dynamice a program tedy neztrácí čas čekáním, ale může hned kontrolovat aktivitu dalších strun.
Po výpočtu dynamiky dojde k volání další pomocné funkce s názvem nota. Té je předáno číslo noty odpovídající MIDI standardu a spočítaná dynamika. Zde také došlo ke změně. Původně byla vyslána zpráva Nota zapnuta a hned poté Nota vypnuta. V případě hraní s pedálem totiž nemá na zvuk vliv, kdy byla nota vypnuta, ale ozývá se, dokud nedojde k puštění pedálu. Tento způsob však nevyhovoval vývojáři software, který by měl problém s časovým překrýváním neboli vrstvením tónů při hraní na stejnou strunu. Shodli jsme se tedy na řešení, že jako první vyšle převodník zprávu Nota vypnuta, čímž případný hrající vzorek ukončí, a až poté bude následovat zpráva Nota zapnuta. Tímto sice nebude nástroj plně kompatibilní s jinými zvukovými generátory, jako jsou například digitální klávesy, ale jeho primární určení je pro skutečné vzorky akustického cimbálu.

7.5 Sustain pedál

Doposud byl popsán způsob snímání strun a následné zpracování dynamiky. K cimbálu však neoddělitelně patří Sustain pedál. Bez něj by nešly rozlišit krátké tlumené tóny od dlouhých netlumených. Pedál u klasického cimbálu pracuje na mechanickém principu. Při jeho stlačení dojde ke zvednutí tlumící vrstvy a struny tak doznívají plynule [1]. V případě elektronického cimbálu však tento pedál nemá se strunami žádnou vazbu. Z hlediska zapojení jde o kontakt, na jehož základě se vysílá MIDI zpráva. Zvukový vliv pedálu se tedy projeví až v software virtuálního nástroje. Jelikož je pedál jeden, využijeme pro něj samostatný vstup čipu. Zde je výhodné použít zapojení s interně vřazeným rezistorem (INPUT PULLUP). Konstruktér zajistí, aby pedál spínal tlačítko. Jeden kontakt tlačítka bude přiveden do čipu na pin s číslem 11 a druhý kontakt tlačítka bude spojen se zemí (0 V). Při sešlápnutí pedálu bude tedy na vstupu logická úroveň LOW. V bloku setup musíme nejprve nakonfigurovat pin č. 11, aby se choval jako vstup s interně vřazeným rezistorem. Nyní je potřeba si uvědomit, jakým způsobem bude samotné vysílání zprávy provedeno. Nabízí se řešení, aby program vyslal MIDI zprávu, když bude na vstupu č.11 logická úroveň LOW.



Obr. 7.6 Schéma zapojení Sustain pedálu

Jelikož ale program běží ve smyčce, tak kdykoli by přišel na místo vyhodnocení došlo by k vyslání zprávy. Délka trvání smyčky je kolem 600 us, takže by se zpráva během jedné sekundy poslala přibližně tisíc šest set krát. Abychom tomuto ději zabránili a nezahlcovali zbytečně čip a MIDI sběrnici, vytvoříme si pomocnou proměnnou, do které budeme zapisovat předchozí stav pedálu. Jelikož pedál může nabývat pouze dvou stavů, využijeme pro uchování této informace datový typ *boolean*, který umožňuje uchovat informaci ve stavu true / false neboli pravda a nepravda. Pokud již bylo vyhodnoceno sešlápnutí pedálu, uložíme do něj informaci pravda. Jakmile bude pedál puštěn, bude tato proměnná přepsána na nepravdu. Abychom mohli proměnnou využívat, musíme ji nejprve deklarovat a nastavit do výchozí hodnoty. Obyčejně se proměnná deklaruje ve funkci, ve které se s ní bude pracovat. V tomto případě to ovšem není možné, protože ve funkci loop by se deklarovala a nastavovala do výchozí hodnoty neustále dokola. Proto ji deklarujeme ještě před funkcí loop a zároveň mimo ostatní funkce. Kdybychom ji totiž deklarovali uvnitř jiné funkce, neměli bychom k ní ve funkci loop přístup. Po deklarování ji nastavíme do stavu false, protože od spuštění programu zatím pedál nemohl být vyhodnocen.

Nyní, když jsme pedálu přiřadili vstup a deklarovali pomocnou proměnnou, můžeme vyhodnocovat jeho stav. K tomu použijeme funkci (if) s dvojitou podmínkou. Zpráva o aktivaci pedálu se vyšle v případě, je-li pedál sešlápnut, tedy je-li na vstupu logická úroveň LOW, a zároveň je-li předchozí stav pedálu false, což znamená že prozatím nebyla vyslána zpráva o jeho sešlápnutí. Pokud jsou obě podmínky splněny, dojde k vyslání MIDI zprávy. Informace je vyslána ve zprávě typu Control Change, která zahrnuje tři *byty*. První *byte* je stavový a určuje typ zprávy. Ve druhém *bytu* je číslo ovládaného kontroleru, což je v našem případě Sustain s číslem 64. Jelikož se jedná o kontroler typu spínač [6], vyšleme ve třetím *bytu* hodnotu 127, která znamená sepnuto. Jakmile je tato zpráva vyslána, přepíšeme pomocnou proměnnou s názvem *sustain* na true. Když tedy program

opět dojde k výše popsané podmínce, nejsou již splněny obě její části a k opětovnému vyslání zprávy nedojde, což bylo naším cílem.
Nyní zbývá zajistit, co se stane v případě uvolnění pedálu. Zde je rovněž cílem, aby byla zpráva vyslána pouze jednou. Opět tedy použijeme podmínku, která vyšle zprávu v případě, že předchozí stav pedálu je true a nyní je na vstupu logická úroveň HIGH. Pokud jsou splněny obě části podmínky, vyšleme MIDI zprávu ve stejném tvaru jako v případě sešlápnutí, akorát ve třetím *bytu* nahradíme hodnotu 127 hodnotou 0. Po vyslání zprávy musíme rovněž přepsat pomocnou proměnnou *sustain* na false, abychom zachovali pravidlo.
V této funkci jsem se také rozhodl použít MIDI zprávu All Notes Off, která se v normě MIDI označuje za povel [6] a spadá pod příkazy Control Change. Vysílají se pouze dva *byty*, protože povely pro svou činnost nepotřebují mít hodnotu, stačí pouze číslo povelu. Povel All Notes Off má číslo 123, které vyšleme v datovém *bytu*. Při zkoušení funkčnosti se ale povel neprováděl, zkusil jsem tedy poslat i třetí datový *byte* s hodnotou 127. Po této úpravě již byl povel vyhodnocen a proveden. Nechal jsem tedy program s touto úpravou.
Z hlediska převodníku jde při vysílání zprávy All Notes Off o redundantní neboli nadbytečnou zprávu, protože informaci o stavu pedálu již vysíláme. Tato zpráva má však význam pro virtuální nástroj nebo sekvencer, který MIDI zprávy zaznamenává. Například u klavíru je jasně definována doba od spuštění po povolení klávesy. U cimbálu jde však pouze o impulz spuštění, který nemá jasně definované ukončení. Aby se dalo lépe orientovat v nahraném MIDI souboru, je tedy dobré po uvolnění pedálu globálně všechny noty vypnout. Samotný Sustain pedál totiž noty nevypíná. U klavíru můžeme po zmáčknutí a držení klávesy sešlápnout a uvolnit pedál třeba třikrát, ale nota bude stále hrát. Toto neplatí u cimbálu, kde se struny při uvolnění pedálu vždy utlumí. Proto je tedy vhodné všechny noty ihned po uvolnění pedálu vypnout.

7.6 Demo skladba

Součástí převodníku je také demo skladba, která bývá také na většině digitálních klávesových nástrojů. Jejím úkolem je předvést, co který nástroj umí. Na doporučení kolegy jsem vybral část skladby Fantazie pro cimbál skladatele Mircea Chiriaca. Z hlediska hardware se jedná o tlačítko se shodným zapojením jako Sustain pedál. Na digitálním vstupu se tedy při stisknutí objeví logická úroveň LOW, na kterou zareaguje podmínka a dojde ke spuštění pomocné funkce s názvem **demo**.

Než jsem tuto funkci vytvořil, bylo potřeba najít způsob, jak do čipu přepsat pořadí not. Jelikož mám tuto skladbu přímo v MIDI souboru, nabízela se řada možností, jak ji naprogramovat do čipu. MIDI soubor je možné otevřít v takzvaném *Score editoru*, který zobrazí skladbu v notové osnově. Zde bych si následně musel ke každé notě dopsat její výšku podle normy MIDI. Další možností je otevřít skladbu v *Key editoru*, který má dvě osy. Osa *x* je časová a osu *y* tvoří virtuální klaviatura. Přepis z tohoto formátu by ale taky nebyl úplně přehledný, protože u jednotlivých grafických vyobrazeních noty není jasně patrný začátek každé z not. Nejkomfortnější přepis nabízel *List editor*. Jde totiž o řádkový seznam MIDI událostí. Každý řádek nese informaci o jedné notě, stejně tak, jako podprogram v čipu. V *List editoru* ovšem není zobrazeno číslo noty, které se po MIDI posílá, zápis je ve tvaru písmene a čísla, například D3. Písmeno značí tón a číslo určuje oktávu. Od hudebního značení se ale liší. Číslo 3 zde totiž nevyjadřuje tříčárkovanou oktávu, ale jednočárkovanou [6]. Referenční číslo 0 totiž označuje kontra oktávu. Názvy not jsem tedy musel převést na čísla not podle tabulky 7.4 [6].

Tab. 7.4 Čísla not podle normy MIDI

Oktáva	Hudební označení	C	C#	D	D#	E	F	F#	G	G#	A	A#	H
-2	sub-subkontra	0	1	2	3	4	5	6	7	8	9	10	11
-1	subkontra	12	13	14	15	16	17	18	19	20	21	22	23
0	kontra	24	25	26	27	28	29	30	31	32	33	34	35
1	velká	36	37	38	39	40	41	42	43	44	45	46	47
2	malá	48	49	50	51	52	53	54	55	56	57	58	59
3	jedno-čárkovaná	60	61	62	63	64	65	66	67	68	69	70	71
4	dvou-čárkovaná	72	73	74	75	76	77	78	79	80	81	82	83
5	tří-čárkovaná	84	85	86	87	88	89	90	91	92	93	94	95
6	čtyř-čárkovaná	96	97	98	99	100	101	102	103	104	105	106	107
7	pěti-čárkovaná	108	109	110	111	112	113	114	115	116	117	118	119
8	šesti-čárkovaná	120	121	122	123	124	125	126	127				

Abych nemusel vypisovat celé MIDI zprávy, použil jsem již vytvořený podprogram s názvem nota, do jehož parametru se zadává výška noty a dynamika. Zde je výhodné, že *List editor* zobrazuje přímo ke každé notě i její dynamiku. Poté, co jsem noty přepsal, zbývalo vyřešit časování, kdy jednotlivé noty přehrát. Tempo skladby je 120 BPM neboli 120 úderů za minutu. Jeden úder se vztahuje na čtvrťovou notu. Trvání této noty vychází na 0,5 s. Úryvek skladby je hraný v triolách, čas k zahrání noty tedy musí být třetinový, což vychází na 166 ms. Mezi jednotlivé příkazy k vyslání not musíme vložit čekání. To se provádí pomocí funkce delay, jejímž parametrem je čas v milisekundách. Pro případnou změnu tempa však není vhodné vypisovat časy do každé instrukce. Vhodnějším řešením je nahradit čas pomocnou proměnnou, kterou vytvoříme na začátku a uložíme do ní hodnotu 166. V případě změny tempa tedy nebude nutné přepisovat několik desítek hodnot, ale pouze jednu. Na závěr demo úryvku je zahrán rozklad s dvojnásobnou rychlostí, čas mezi notami bude tedy poloviční a to 83 ms. Opět bude uložen ve vlastní proměnné. Programu sice zabere nějaký čas na vyslání noty, tento čas je však v řádu desítek mikrosekund, proto jej můžeme zanedbat. Tempo se bude lišit maximálně o dvě setiny procenta, což je nepostřehnutelná změna.

Při stisku Demo tlačítka se tedy provede následující seznam instrukcí. Nejprve dojde k vytvoření a nastavení proměnných pro dobu čekání mezi notami. Poté se přehraje první nota a program čeká stanovenou dobu, než přehraje druhou notu. Tak to jde až do místa, kdy je vyslána MIDI zpráva o změně stavu pedálu na zapnuto. Zpráva se skládá ze třech *bytů,* jak bylo již uvedeno v kapitole 7.5. Dále dojde k přehrání not s dvojnásobnou rychlostí. Po poslední notě program čeká čtyři takty, což je 2 656 ms, než noty dozní a vyšle zprávu o změně stavu pedálu na vypnuto. Následně je vyslána MIDI zpráva All Notes Off, kdy dojde k vypnutí všech not. Po skončení Demo skladby je opět prováděn hlavní podprogram s názvem loop, který opět kontroluje aktivitu strun.

8 ZÁVĚR

Práce byla pojata jako postup, jak navrhnout a zkonstruovat MIDI převodník. V úvodu bylo nastíněno, jak bude návrh probíhat a jaké jsou jeho kroky. Ve druhé kapitole byl stručně probrán akustický hudební nástroj cimbál, z čeho se skládá a jaké prvky slouží jako excitátor, oscilátor, rezonátor a radiátor.

Ve třetí kapitole byly popsány různé způsoby snímání strun s jejich následným převodem do digitální podoby. Byly také popsány jejich výhody a nevýhody. Z těchto druhů snímání byl vybrán nejvhodnější, a to snímání na principu tlačítek.

Ve čtvrté kapitole byl popsán protokol MIDI, jaké používá konektory, jakým způsobem a jakou rychlostí komunikuje. Dále popis zpráv, které se pomocí tohoto protokolu dají přenášet a také detailní popis struktury datového rámce.

Pátá kapitola se věnovala výběru vhodného jednočipu, co vše by měl splňovat a obsahovat. Dále byl popsán vybraný jednočip Atmel ATmega 2560, použitý pro konstrukci převodníku.

Šestá kapitola obsahovala návrh MIDI převodníku. Je zde prodiskutováno řešení zapojení, způsob rozšíření pro efektivnější využití vstupů a zajištění kompatibility s jinými virtuálními nástroji. V rámci této kapitoly byl vytvořen prototypový program pro snímání tlačítek stupnice C dur s určením dynamiky na základě časového rozdílu mezi sepnutím dvou tlačítek. Program je podrobně popsán a je vysvětlen princip převodu a vysílání MIDI zpráv.

V sedmé a zároveň nejrozsáhlejší kapitole byl program rozšířen pro potřeby cimbálu. To obnášelo rozšíření počtu vstupů, úpravu vstupních signálů a samotné rozšíření kódu pro všech 58 strun. Následovalo odzkoušení a vyladění časových konstant pro určování dynamiky, které byly nakonec nastaveny na 0,4 ms pro nejsilnější úder a 20 ms pro nejslabší úder. Dále bylo navrženo a vyzkoušeno zapojení Sustain pedálu s naprogramován jeho obslužné funkce. Poslední doplňující funkcí byla funkce demo, která ukazuje, že převodník nemusí umět data jen převádět, ale může je také sám přehrávat.

V rámci budoucího rozšíření nástroje je plánováno přidat ovládací prvky, jako je například přepínač typů paliček. Na základě přepínače by se tak posílaly MIDI zprávy typu Control change, které by virtuální nástroj přijímal a podle nich načítal do paměti RAM a následně přehrával vzorky pro různé druhy paliček.

Literatura

[1] HOLCOVÁ, Kateřina. *Konstrukční vývoj cimbálu na Moravě*. Olomouc, 2014. Bakalářská práce. Univerzita Palackého v Olomouci.

[2] VŠIANSKÝ, Pavel. *Cimbály Všianský* [online], 2008, dostupné z: http://cimbaly.cz/

[3] Procházka, Petr. *Procházka Custom Guitars* [online], 2008, dostupné z: http://www.guitar-makers.com/www/snimani.php

[4] MACHAIN, Emil. *Vývoj a možnosti elektronických bicích nástrojů.* Brno, 2015. Diplomová práce. Janáčkova Akademie Múzických umění v Brně.

[5] MORRISSEAU, Franck. *LaserHarp* [online], 2002, dostupné z: www.harpelaser.com

[6] SCHIMMEL, Jiří. *Komunikační rozhraní MIDI* [online], 2002, dostupné z: http://www.elektrorevue.cz/clanky/02069/index.html

[7] GUÉRIN, Robert. *Velká kniha MIDI,* 2004, ISBN 80-7226-985-2

[8] Roland RD-700NX. Roland [online]. Dostupné z: https://www.roland.com/us/products/rd-700nx/

[9] Arduino MEGA 2560 [online], 2016, dostupné z: http://www.arduino.org/products/boards/arduino-mega-2560

[10] VRBA, Kamil, Norbert HERENCSÁR a Jaroslav KOTON. Analogová technika. Brno, 2011. Skripta. Vysoké učení technické v Brně.

[11] VRBA, Kamil. Konstrukce elektronických zařízení: Aplikační pravidla pro aktivní elektronické obvodové prvky. Brno, 2017. Skripta. Vysoké učení technické v Brně.

[12] Multiplexer CD74HCT151. Alldatasheet.com [online], 2003, dostupné z: http://www.alldatasheet.com/datasheet-pdf/pdf/27012/TI/CD74HCT151.html

Seznam symbolů, veličin a zkratek

A/D	-	Analogově digitální převodník
DIN 5	-	Typ konektoru podle Německé národní normy
kBaud	-	kilo Baud, jednotka přenosové rychlosti
kV	-	kilo Volt, jednotka napětí
mA	-	mili Ampér, jednotka proudu
MHz	-	mega Hertz, jednotka frekvence
ms	-	mili sekunda, jednotka času
Ω	-	Ohm, jednotka elektrického odporu
us	-	mikro sekunda, jednotka času
USB	-	Univerzální sériová sběrnice
V	-	Volt, základní jednotka napětí

Seznam příloh

A OBSAH PŘILOŽENÉHO DVD

Přiložené DVD obsahuje:

- Matěj Korytář – bakalářská práce.pdf – elektronická verze bakalářské práce
- MidiConverterCdur.ino – program pro stupnici C dur ve formátu Arduino file
- MidiConverterCdur.pdf – program pro stupnici C dur ve formátu pdf
- MidiConverterCimbal.ino – rozšířený program pro cimbál ve formátu Arduino file
- MidiConverterCimbal.pdf – rozšířený program pro cimbál ve formátu pdf